Effects of Graph Augmentation of Graph Neural Network Twitter Bot Detection Models

Juan Sanchez Moreno and Karen Works
Department of Computer Science
Florida State University
Panama City, FL 32405
jfs22b@fsu.edu, keworks@fsu.edu

1 Introduction

There is a growing concern about the presence of bots in social media, especially on Twitter (now X). These bots can spread misinformation, impose narratives, and distort the reality of the platform's users. One current approach to identifying bots utilizes Graph Neural Networks (GNN). The goal of our research is to identify a GNN that is both accurate and efficient. These two constraints can be competing forces. To increase accuracy, often requires more data which leads to longer processing times. To increase efficiency, often requires less data to be processed which leads to a decrease in accuracy.

2 Dataset and Baseline Model

TwiBot-20 [3] is the "big data set" used in Twitter bot detection GNN model. For our research we used TwiBot-22 [2], the latest version of TwiBot which has more features between data points. We seek to improve the BotRGCN [3] model. Hence we started by creating the GNN outlined in BotRGCN [3] model. Our results were sufficiently close to the original BotRGCN model [2] and serve as the baseline for our experiments.

3 Results and Discussion

We first located the combination of relationships that yielded the best inference accuracy, which was the combination of the followers and the quoted relationships. The test accuracy of this relationship pair was 0.7802 which is approximately 0.5% higher than our baseline.

Given the success of DropMessage [1], we then explored dropping edges. Edge dropping has been shown to produce a more robust model [1]. In addition, it requires less CPU time to make predictions on a reduced graph. We measured efficiency as the time it took a model to make inferences on the full dataset (training + testing) 200 times. We found the best-performing random edge dropping of both accuracy and inference time at 5% which reported an average completion time of 18.3 s with a test accuracy of 0.7256.

Then, we considered taking a more planned approach to select edges to drop from nodes that have at least 4 edges already. Our goal is to prevent loss of connectivity to any edge. This technique revealed promising results. The model's average inference accuracy peaks at 0.7730 with 3% edge reduction. This is close to our baseline with no edge drops resulting in a 0.19% lower inference accuracy than our baseline. However, there was an inference CPU time efficiency increase. The best-performing value of edge dropping (3%) reported an average completion time of 18.21 s on the test while the model with the lowest dropping value (0.01%) reported an average of 18.5 s. This is a 0.29 second difference.

4 Conclusion

We successfully built a GNN with a combination of relationships that produced a higher prediction accuracy than our baseline and edge dropping that decreased the CPU execution time. The graphs created from social network data are massive and constantly change to reflect real time relationships. Supporting these requires expensive computations to train and make predictions running these models continuously. In the future, the development of a real time efficient edge-dropping technique could be used to reduce this overhead.

References

[1] Taoran Fang, Zhiqing Xiao, Chunping Wang, Jiarong Xu, Xuan Yang, and Yang Yang. Dropmessage: Unifying random dropping for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4267–4275, 2023.

[2] Shangbin Feng, Zhaoxuan Tan, Herun Wan, Ningnan Wang, Zilong Chen, Binchi Zhang, Qinghua Zheng, Wenqian Zhang, Zhenyu Lei, Shujie Yang, et al. Twibot-22: Towards graph-based twitter bot detection. *Advances in Neural Information Processing Systems*, 35:35254–35269, 2022.

[3] Shangbin Feng, Herun Wan, Ningnan Wang, Jundong Li, and Minnan Luo. Twibot-20: A comprehensive twitter bot detection benchmark. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 4485–4494, 2021.

Evaluating Parallel Processing Using Merge Sort

Sam Bostian, Michael Rizig, Jonathan Turner, Eli Headley, Charlie McLarty, Ernesto Perez and Daron Pracharn
Instructor - Dr. Patrick Bobbie
Kennesaw State University
Marietta, GA
sbostian@students.kennesaw.edu

***Abstract*—Multicore processing offers the advantage of dividing and sharing computer resources among interconnected processes, mitigating bottlenecks and minimizing wasted potential caused by idle computing hardware. Given the substantial computational demands of such problems, parallelizing and distributing computing tasks across multiple cores is often more cost-effective than relying on a single powerful processor. However, one drawback of multicore processing lies in the complexity of coordinating computer resources. The objective of this project is to leverage parallelization to sort data using an implementation of Merge Sort. The approach for this project involved establishing a multithread pool and utilizing the Single Program Multiple Data (SPMD) model. Comparisons were made with the speedup, efficiency, and runtimes achieved by increasing the number of distributed cores across different array sizes against the metrics of a single-core processor.**

Keywords—Threading, Multithreading, Merge sort, Parallel Computing

Introduction

This experiment aims to answer the question: "How significant would the performance increase if the datasets become exponentially larger for each increase in the number of threads used for processing?" An attempt to answer this question by increasing each of the data sets using arrays with sizes of 10,000, 100,000, 200,000, and 300,000. Each array was populated with random and distinct integers ranging from 1 to 999,999. Subsequently, the arrays were transmitted via a master thread to a thread pool. The thread pool then executes a merge sort on the divided components on the array then passes the results to the master thread to reassemble the array. The results obtained for each scaled array size were compiled into a table and graphed to analyze the results.

Data Analysis

From each experiment the time was collected, in nanoseconds. The timing begins at the creation of the threads and finishes when the data from each thread finishes merging creating a complete sorted array. For the experiments the improvement in performance of increasing the number of parallel processors versus the serial one, is measured using the speedup and efficiency metrics. The speedup, the ratio of the program runtime in serial over the runtime in parallel:

$$Speedup(n,p) = \frac{T_{Serial(n)}}{T_{Parallel(n,p)}} \quad [1]$$

Where n is the size of the input and p is the number of processors. A perfect speedup score is where the speedup equals the number of processors, Speedup(n,p) = p, also known as linear speedup. To determine how each processor contributed to the speedup parallel efficiency is used. Parallel efficiency is calculated using the following formula:

$$Efficiency(n,p) = \frac{Speedup(n,p)}{p} = \frac{T_{Serial(n)}}{p * T_{Parallel(n,p)}} \quad [1]$$

Parallel efficiency is given by the speedup over the number of processors.

Results

As the array size increased the benefits of parallel processing can be seen in Table 1. When the array is only 10,000 the time it takes to process the array is approximately half from one to two and from two to four processers. When eight and sixteen processors are used on an array of 10,000 elements the decrease in time to process the arrays is hampered by the work to divide the array. The runtime decreases between about three quarters when the number of processors is increased between two and four processors. When the processors are increased to eight or sixteen the runtime is only decreased by about half.

Runtime				
Number of Threads	Array Sizes (# of elements)			
	10000	100000	200000	300000
p = 1	591	48221	183908	403233
p = 2	258	14191	57020	109616
p = 4	123	4532	15974	34549
p = 8	123	2278	8335	18565
p = 16	104	1257	4668	10450

Table 1: Runtimes for the randomly generated arrays.

While an array of 10,000 elements might look like it does not decrease but that is due to the fact that the 300,000 elements decrease so much compared to the 10,000 elements array.

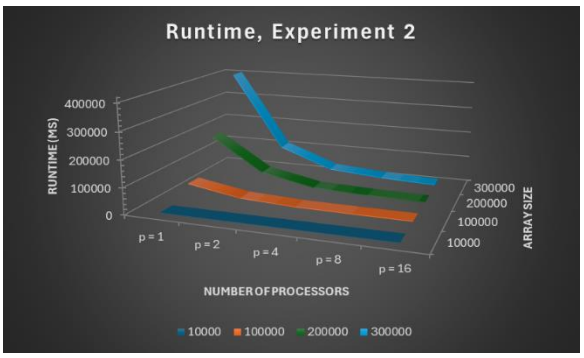


Figure 1: A line graph displaying the recorded runtimes for each array size for a given number of processors.

The speedup gained by using parallel processing is almost equal for each processor regardless of the size of the array. The speed up almost increase by the same factor has the number of processors doubles. The speed up is more than tripled from one processor to two processors and from two to four processors. The speedup increases of a factor of two when the number of processors increases from four to eight and then from eight to sixteen the speedup is only gained by a factor of 1.75.

Speedup				
Number of Threads	Array Sizes (# of elements)			
	10000	100000	200000	300000
p = 1	1	1	1	1
p = 2	3.68	3.4	3.23	3.68
p = 4	11.67	10.64	11.51	11.67
p = 8	21.72	21.17	22.06	21.72
p = 16	38.59	38.36	39.4	38.59

Table 2: Calculated speedup for the randomly generated arrays.

The graph in figure 2 shows that the speedup follows the same shape as the theoretical big O trajectory as a merge sort, which is $O(n\log_2 n)$ [2].

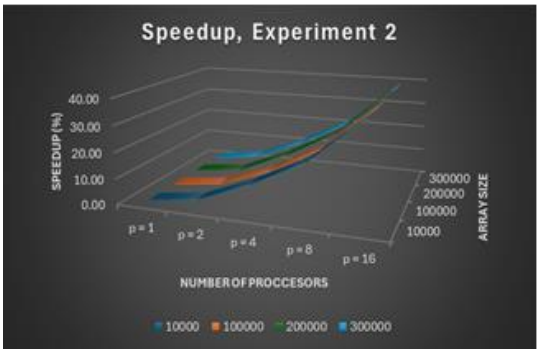


Figure 2: A line graph displaying the calculated speedup for each array size for a given number of processors.

Unlike the speedup the efficiency did increase the same amount for about all the array sizes. The 10,000-element array followed the same pattern as the larger arrays but had a lower increase in efficiency for each increase in the number of processors used. All the arrays had their greatest efficiency when four threads were used to sort an array. The efficiency for each processor begins to decrease after four threads are used.

Efficiency				
Number of Threads	Array Sizes (# of elements)			
	10000	100000	200000	300000
p = 1	1	1	1	1
p = 2	1.15	1.7	1.61	1.84
p = 4	1.2	2.66	2.88	2.92
p = 8	0.6	2.65	2.76	2.72
p = 16	0.36	2.4	2.46	2.41

Table 3: Calculated efficiency for the randomly generated arrays.

The changes in efficiency for each array are visualized in figure 3. The arrays 100,000 elements and larger the slope of the graph increases between a factor of 0.6 to 0.8 for one to eight processors used. When the number of processors is increased to eight and higher the efficiency is between 88% to 95% the efficiency of the previous number of processors. The ten thousand element array has the smallest gain in efficiency and has the largest decrease in efficiency as the number of processors increases after four threads. Efficiency only increases by 15% and then 5% as the number of processors is doubled from one to four. When the number of processors is doubled again to eight and sixteen the efficiency decreases by about 50% for each increase.

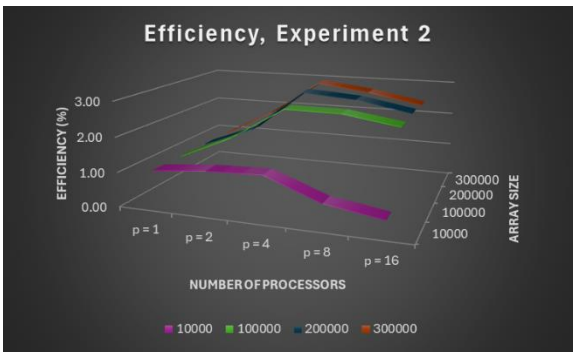


Figure 3: A line graph displaying the calculated efficiency for each array size for a given number of processors.

Conclusion

The primary goal of this research paper was to simulate parallel processing. Using SPMD task parallelism method of implementation effectively addressed the issues of multi-threading including thread synchronization and load balancing of the data among the concurrently running threads [1].

For the large arrays (>100,000), it was observed that a significant speedup occurred when the number of threads increased. This can be explained due to the cost of overhead minimal compared to the increased efficiency of the added cores. The efficiency in these large test cases indicated a speedup where the speedup is greater than anticipated for an increased number of cores.

This research showed that there is no optimal number of cores that will suit all cases. To allow more consistent results in a dynamic environment of differing input sizes, a threshold could've been implemented to assign the number of cores on runtime. Overall, it was determined that spending the extra time to implement parallel processing for a sorting algorithm yielded significantly better results when the amount of data is substantial.

References

[1] P. Pacheco, "An Introduction to Parallel Computing", Elsevier Inc., 2011. ISBN-10: 01237426095

[2] A. Levitin, "The Design and Analysis of Algorithms", Pearson, 2012. ISBN-10: 0-13-231681-1

Power Naps to Improve the Performance of Consolidated Learning

Skyler Gipson
Stetson University
Faculty Advisor: Dr. Hala ElAarag
Department of Math and Computer Science

Abstract

In the context of training deep neural networks (DNNs), continual learning methods are a contender in reducing model inaccuracies that result from a model "forgetting" previous knowledge learned when updated offline, called catastrophic forgetting. While recent approaches to improving the performance and memory retention of deep neural networks have been inspired by cognitive neuroscience and modeling of the human brain, current memory models must be further developed to increase the efficiency and performance of deep neural networks by limiting how susceptible they are to catastrophic forgetting. In humans, memory consolidation has been found to be related to sleep phases, including NREM and REM sleep, dreaming, and napping. Models utilizing a computational framework based upon wake-sleep phases have outperformed state-of-the-art models, but further work is needed to fine-tune current approaches to memory retention, decay, and interference modeling in continual learning methods [1].

While the wake sleep consolidated learning model proposed in Sorrenti *et al.*, 2023 is novel and successful in improving the performance of continual learning methods, it does not acknowledge power naps as being an influential aspect of memory consolidation, which occurs through neocortical-hippocampal interaction during NREM sleep [2, 3]. We ultimately seek to incorporate offline-brain states, specifically napping, into the deep neural network training process to reduce catastrophic forgetting, in the context of continual learning.

Methodology

We plan to base our strategy on a 24-hour wake-sleep cycle for a human, modifying the framework proposed in Sorrenti *et al.*, 2023, shown in *Fig. 1*. The DNN emulates the neocortex,

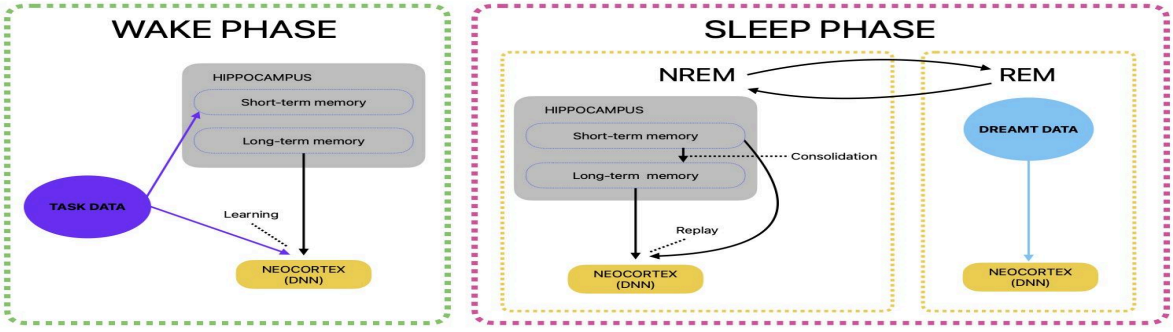


Figure 1: **DNN Wake-Sleep Consolidated Learning**

in that connections are updated based on new and previous information, these being task data and long-term memories from previous tasks, respectively. In the wake phase, a single data stream for a task mimics an experience, stored as an episodic memory in short-term memory to later be

replayed and consolidated into long-term memory during NREM sleep. During REM sleep, the model is exposed to a datastream sharing no characteristics with any task data, previous or current. As memory consolidation occurs, hippocampal-dependence decreases such that the short-term memory of our wake-sleep framework can effectively be wiped, since its contents have been transformed into knowledge in the DNN and stored in long-term memory [4]. This step restores the encoding capacity of the hippocampus for the next wake period. However, the long-term memory buffer implemented in Sorrenti *et al.* (2023) is still required, given that while the DNN receives new information there is potential for catastrophic forgetting as its synaptic connections are updated. Therefore, the contents of short-term memory must move to long-term memory, which in turn is passed into the DNN (the neocortex) alongside task data, ensuring that model parameters are updated considering all existing knowledge as well as the new task characteristics.

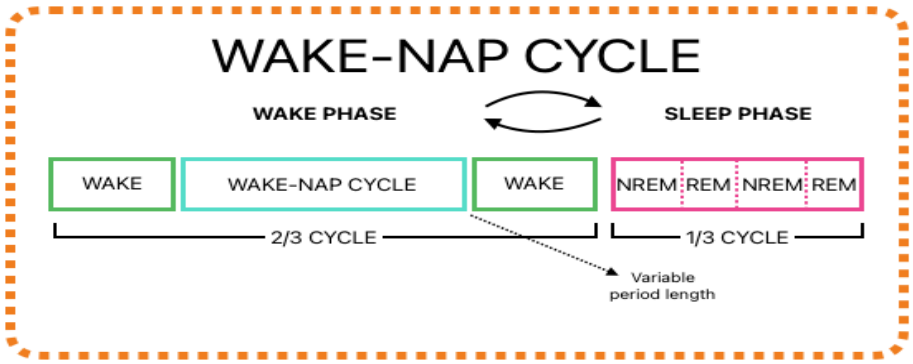


Figure 3: **Consolidated Learning Strategy Through Power Nap Periods**

Given that the hippocampus has restored encoding capacity after memory consolidation, various possibilities exist in how power naps could improve model performance, efficiency, and reduce catastrophic forgetting. Implementing multiple, shorter NREM sleep phases to break up the wake phase into multiple wake phases–each containing only a portion of data to be learned for the 24-hour cycle–would allow for fewer "experiences" being held in short-term memory to be reviewed during the subsequent, post-learning nap. Thus, during a nap fewer task experiences will be replayed and consolidated during NREM sleep than during a full night's sleep, given that the model was exposed to less task data in the preceding wake phase.

Conclusion

We hypothesize that separating the singular wake phase implemented in Sorrenti *et al.* (2023) into a wake-nap cycle will improve both model performance and memory storage, supposing that short-term memory information stored during a nap is learned more deeply, as synaptic connections are updated based on a smaller amount of information. No method to date has yet incorporated power naps as a driving factor in updating deep neural networks. As naps have been proven to play a role in memory and learning, neuroscience-based CL methods would benefit from an approach incorporating these.

References

1. Sorrenti, Amelia, et al. "Wake-Sleep Consolidated Learning." *arXiv preprint arXiv:2401.08623* (2023).

2. Lau, H., M. A. Tucker, and W. Fishbein. "Daytime napping: Effects on human direct associative and relational memory." *Neurobiology of learning and memory* 93.4 (2010): 554-560.

3. Sirota, Anton, et al. "Communication between neocortex and hippocampus during sleep in rodents." *Proceedings of the National Academy of Sciences* 100.4 (2003): 2065-2069.

4. Saletin, Jared M., and Matthew P. Walker. "Nocturnal mnemonics: sleep and hippocampal memory processing." *Frontiers in neurology* 3 (2012): 59.

ANALYZING STUDENTS' COLLABORATION PATTERNS VIA A GRAPH VISUALIZER

Jiabao Xu, Zhijun Zhao
Advisors: Abdussalam Alawini
University of Illinois at Urbana-Champaign

1 Introduction

Collaborative learning has long been a key teaching paradigm in education. Traditionally, educators have assessed collaborative behaviors using indirect measures such as test scores or qualitative feedback [1,2]. However, the increasing adoption of computer-based tools in classrooms offers the opportunity to directly observe and analyze student collaboration patterns in real time with minimal intervention. In this context, our study presents GA2GRAPH, a graph-based system that enables educators to monitor and analyze student collaboration through submission data, offering a more immediate and detailed understanding of these interactions.

GA2GRAPH is a full-stack pipeline that converts students' submission records into a graph structure. The system consists of a graph-conversion process, a Neo4j backend to store collaboration data, and a frontend interface that allows educators to visualize and analyze collaboration patterns with a range of filtering options. This integrated system offers an efficient and flexible way to gain insights into student teamwork and collaboration dynamics.

This paper presents our findings on student collaboration patterns in a university Database Systems course, analyzed using GA2GRAPH.

2 METHODS

2.1 Data Collection

Student collaboration data was collected and anonymized from the Database Systems course at the University of Illinois at Urbana-Champaign during the Fall 2022 (514 students) and Spring 2023 (470 students) semesters. This upper-level computer science course is offered to both undergraduate and graduate students and is taught using a flipped classroom model. In this model, the instructor delivers the course material through pre-lecture videos, while in-class time is dedicated to collaborative group work. Students form teams of up to four members to complete Group Activities (GA) on the PrairieLearn platform, applying the knowledge taught in the videos.

The GA content was consistent across both semesters and covered a broad range of database topics, including SQL, Database Design, Indexing and Storage, Transaction Management, Query Processing and Optimization, MongoDB, and Neo4j. Each GA featured a variety of question types tailored to the specific topic, such as auto-generated coding exercises, relationship-drawing tasks, multiple-choice questions, and short-answer questions.

2.2 Tool Development

Graph Conversion: We initially obtained raw submission data logs in JSON format from PrairieLearn, which we then parsed to extract both the submission content and the corresponding feedback. After cleaning the data, we represented each student as a node and mapped consecutive submission records to a GA question as edges. If a student submitted two consecutive answers, a self-directed edge was created. Each node includes features such as anonymized group names, student IDs, and collaboration patterns, while edges are enriched with properties like submission occurrence count and the average time interval between submissions. The structured data was then stored in a graph database Neo4j.

GA2GRAPH Interface: The GA2GRAPH interface is connected to the Neo4j database via Apollo Server/Client, enabling fast demonstration and advanced filtering of the collaboration graph. The interface visualizes student collaboration patterns within a single GA for a specified semester. In this visualization, each node represents a student, and edges illustrate the collaboration between two students. Each subgraph corresponds

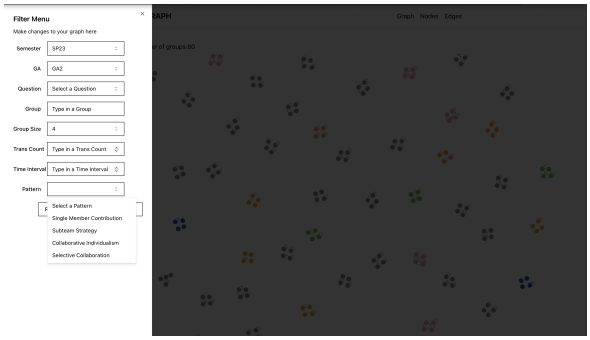


Figure 1: The GA graph of semester 2023-GA2 filter panel open

to a student group, with group sizes ranging from 1 to 4.

A filter panel on the left allows users to switch between semesters and specific GAs. It also provides advanced filtering options: the 'Question' filter enables users to display only nodes and edges related to a selected question, while the 'Time Interval' filter shows the average time spent on submissions between pairs of students for the current GA.

Clicking on a node or edge reveals a details panel on the right, which displays additional features of the selected nodes and edges, along with a subgraph visualization of the student's group. This feature allows for a closer examination of collaboration patterns.

3 Result

Through analysis of the collaboration graphs in GA2GRAPH, we identified four primary types of collaboration patterns: Single Member Contribution, Subteam Strategy, Selective Collaboration, and Collaborative Individualism.

Single Member Contribution: All the answers were submitted by a single student in the group, indicating a strong leadership role, with other members contributing minimally.

Subteam Strategy: The group was divided into smaller subteams, where students collaborated primarily within their subteam partners, resulting in limited cross-team collaboration.

Selective Collaboration: Students actively collaborated across multiple questions, engaging with different group members on various tasks, resulting in each student submitting to multiple questions.

Collaborative Individualism: Students divided the GA into separate questions, with each

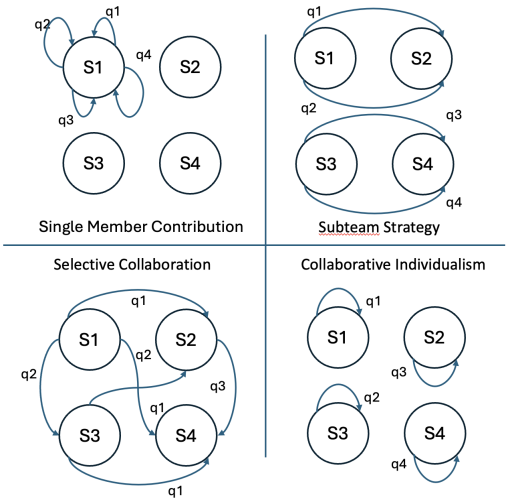


Figure 2: Four different collaboration patterns

member focusing primarily on one question, working independently from the rest of the group.

These patterns were observed consistently across different GAs. Notably, the same group often adopted different collaboration patterns depending on the GA question type. For instance, when applying the GA filter, we noticed that in GAs with coding questions, groups tended to employ more diverse collaboration strategies. Conversely, in GAs with multiple choice or short answer questions, where the questions are refreshed upon incorrect submissions, students frequently adopted Subteam or Selective Collaboration strategies, collaborating more intensively to complete the tasks.

4 Conclusion

In this study, GA2GRAPH revealed distinct collaboration patterns among students in a Database Systems course. By visualizing submission data, educators can gain valuable insights into group dynamics and adapt teaching strategies to enhance collaborative learning across varied activities.

References

Jade McKay and Bhavani Sridharan. 2024. Student Perceptions of Collaborative Group Work (CGW) in Higher Education. *Studies in Higher Education*, 49(2), 221–234.

Bin Sha, Xiaoyu Gu, Qinfang Zhong, Houren Xiong. 2023. Impact of Peer Learning on the Academic Performance of Civil Engineering Undergraduates: A Case Study from China. *International Journal of Engineering Education*, 39(6), 1417–1433.

"Designing Chatbots for Enhanced Website Navigation"

Ayham Makhamra, amakhamra@mail.roanoke.edu, Faculty Mentor: Dr. Adewale Sekoni, sekoni@roanoke.edu
Roanoke College, 221 College Lane, Salem, VA 24153

Introduction

In today's digital age, efficient and quick website navigation is essential for users seeking to extract valuable information from extensive text and content. Chatbots, powered by AI and Natural Language Processing (NLP), offer a solution to this problem by guiding users through websites with ease. This research presents a comprehensive framework for developing custom chatbots based on large language models (LLMs). LLMs, trained on extensive datasets, excel at understanding and generating human-like text, enabling chatbots to efficiently process user queries and retrieve relevant information. The study explores three distinct approaches for designing chatbots to enhance website navigation: rule-based system, fine-turning LLMs, and retravel-augmented generation (RAG).

Methodology

For this research, three approaches were investigated:

Approach 1: Rule-Based Chatbot.

Using Python's Chatterbot library, a basic rule chatbot was built. Rule-based systems use predefined rules (categorized corpus) to respond to user queries and it performs well for straightforward question-and-answer (Q&A). However, the chatbot is limited by its ability to handle complex queries or learn from interactions. This approach can be suited well for static and repetitive tasks, but not dynamic user input.

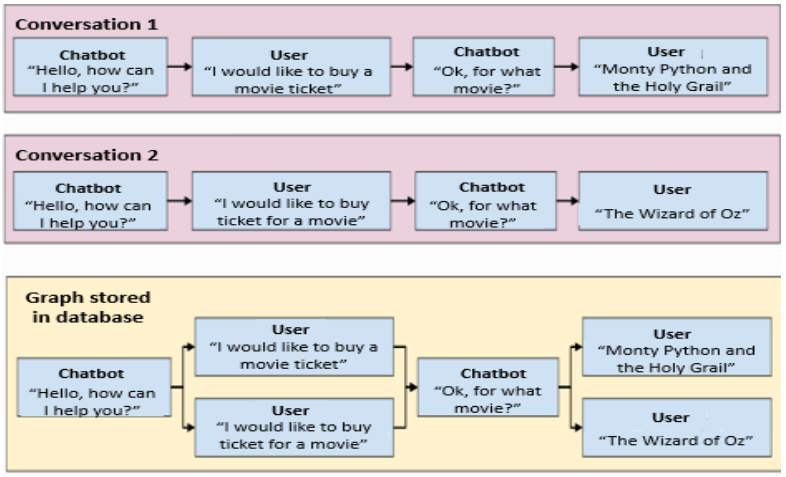


Figure 1. **Chatterbot Graph**

Approach 2: Fine-Tuning Large Language Models (LLMs).

This approach utilizes the "Gemma-2b-it model" along with HuggingFace APIs to fine-tune pre-existing LLM. This process involved training the model on specific datasets (small coding datasets in our case). This method allows the chatbot to generate contextually accurate and domain-specific responses resulting in notable difference compared to the base model, showcasing fine-tuning success. This approach is limited in cases where real-time information is required, as the models are only as current as their training data.

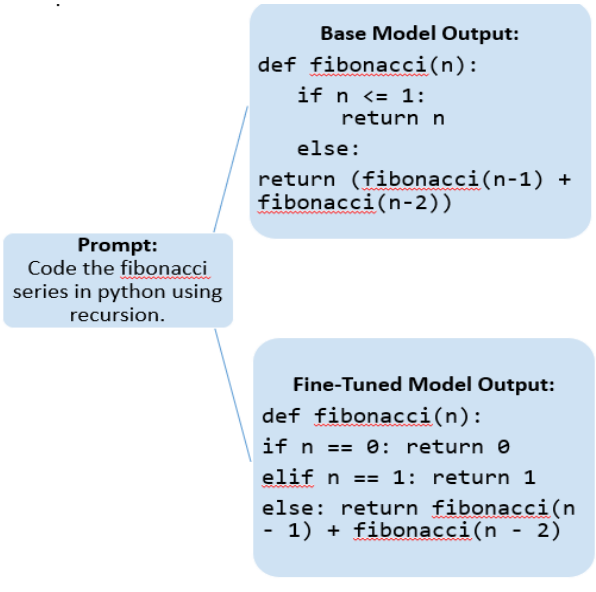


Figure 2. **Gemma Model**

Approach 3: Retrieval-Augmented Generation (RAG)

RAG models, a hybrid solution, combines LLMs with external knowledge sources. RAG trains models by retrieving data from external sources in real-time. This hybrid approach enhances accuracy, and relevance of chatbot responses, making it ideal for integrating real-time or up-to-date information into navigation. The model offers highly adaptable and dynamic solution for the website navigation.

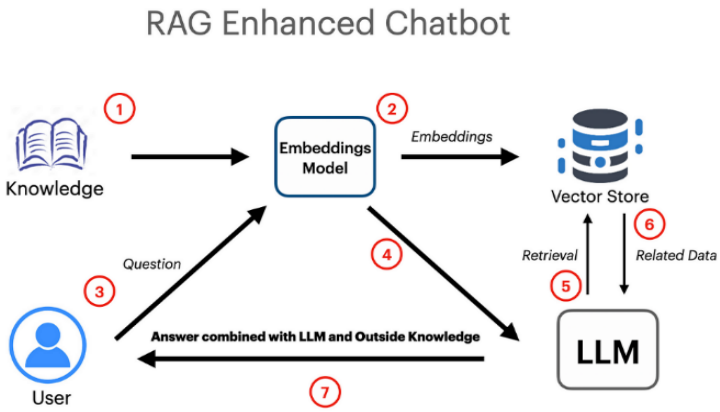


Figure 3. **Retrieval-Augmented Generation**

Results

The performance of the three chatbot systems was evaluated depending on their ability to handle website navigation tasks:

- **Rule-Based System**: This chatbot demonstrated limited functionality and adaptability. While it was efficient for handling predefined tasks, it was unable to adapt to user input beyond its trained rules. Its functionality can be useful for basic navigation but lacks contextuality.
- **Fine-Tuned LLM (Gemma-2b-it):** This model showed a significant improvement in domain-specific questions and the ability to deliver contextually accurate responses. The fine-tuning process enabled the chatbot to better understand the nuances of specific queries, but it was limited while addressing real-time information needs.
- **RAG Model:** The RAG model is expected to dynamically improve website navigation by integrating real-time data retrieval with generative models. This represents a promising solution for large-scale, data-intensive websites that require static and up-to-date information integration.

Future Work

Continue the development of the RAG model for integration into large-scale websites such as the college's website.

Acknowledgements

This research was supported by the MCSP Department. Special thanks to Dr. Adewale Sekoni for his guidance and feedback throughout the project.

References

[1] Ł. Kaiser, E. Shyu, Y. B. Mourri, "Natural Language Processing Specialization," Coursera, https://www.coursera.org/specializations/natural-language-processing, Accessed on: April 12, 2024.

[2] Pokhrel, Sangita & Ganesan, Swathi & Akther, Tasnim & Mapa Senavige, Lakmali Shashika Karunarathne. (2024). Building Customized Chatbots for Document Summarization and Question Answering using Large Language Models using a Framework with OpenAI, Lang chain, and Streamlit. Journal of Information Technology and Digital World. 6. 70-86. 10.36548/jitdw.2024.1.006.

[3] Ebsen, Ty & Segall, Richard & Hyacinthe, Aboudja & Berleant, Daniel. (2024). A Customer Service Chatbot Using Python, Machine Learning, and Artificial Intelligence. Journal of Systemics, Cybernetics and Informatics. 22. 38-46. 10.54808/JSCI.22.03.38.

Imbalanced Data Classification Using Supervised Variational Autoencoders

Clayton McLamb

Elon University

Faculty Advisor: Dr. Scott Spurlock

Introduction

This project focuses on the development and utilization of deep learning models to increase machine learning (ML) fairness, specifically applied to tabular data. Increasingly, ML models play a crucial role in decision making, from loan approvals to medical diagnoses.

However, sampling and representation bias can lead to datasets with class imbalance, where there are significantly fewer examples of minority classes (e.g., disease is present) compared to majority classes (disease is not present). Models trained using this data often develop an implicit bias towards the majority class, failing to accurately predict the minority class [1].

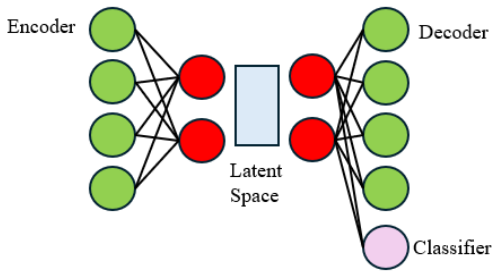
Current algorithms attempt to re-balance the data, either by removing majority examples (undersampling) or by generating new minority examples (oversampling). Although these algorithms can decrease the bias within these models, they often lead to overfitting or generate examples that fail to add any "new" information. This research investigates the capabilities of deep learning to generate new examples, improve model fairness and performance, and match the complex structure of the data.

Methods

One type of deep learning model, a variational autoencoder (VAE), has proven generative capabilities. VAE's consist of two parts, an encoder and a decoder [2]. The encoder learns to deconstruct the data into a lower-dimensional and non-linear space, often referred to as the latent space. The latent space of a variational autoencoder captures the key or salient features of the data, as principal component analysis (PCA) does in a linear space. The decoder uses the compressed version of the data within the latent space for reconstruction.

This research investigates an alternative, supervised variational autoencoder (sVAE). Attached to the hidden layer of the decoder, a classifier predicts what class an observation may belong to. Compared to a VAE, which takes into account purely reconstruction and KL-divergence, the sVAE is capable of identifying key features that separate the classes. Figure 1 displays the architecture of the sVAE.

Figure 1. Architecture of Supervised VAE



When trained on imbalanced data, the sVAE will learn how to generate and classify both minority and majority data. The latent representation within the sVAE will capture features that both minimize reconstruction error, but also identify features that separate class labels. Using this latent representation, sampling will occur to reconstruct new minority examples. The sampling technique investigated here will be compared across a variety of imbalanced datasets and sampling techniques, such as SMOTE, oversampling, undersampling, etc.

Results

Currently, the sVAE has been evaluated on two key features, its reconstruction capability and latent representations. While traditional VAE's exclusively emphasize reconstruction error and KL-divergence, sVAE's also take into account classification. The VAE's unawareness of class label results in heavy emphasis on reconstruction capability and a latent representation that does not intentionally separate classes. The reconstruction capabilities of the VAE and sVAE were tested on the popular imbalanced dataset *ecoli*. Figure 2a displays the reconstruction data using the principal components of the original data. Both models learn to reproduce the original distribution of the input data.

The second capability of the sVAE examined is the latent representation of the data. In order to generate quality minority examples, the minority class must have a defined space within the latent representation to sample from. Figure 2b displays the latent representation of the *ecoli* dataset for both the VAE and sVAE. Compared to the VAE, the sVAE latent representation for minority class data was more defined. By comparing the latent representations across many datasets, we can measure the success of the sVAE. The project and findings will be presented at the conference.

Figure 2a. Comparing Reconstruction

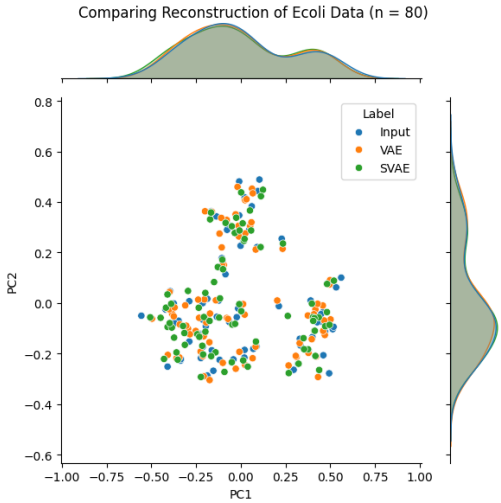
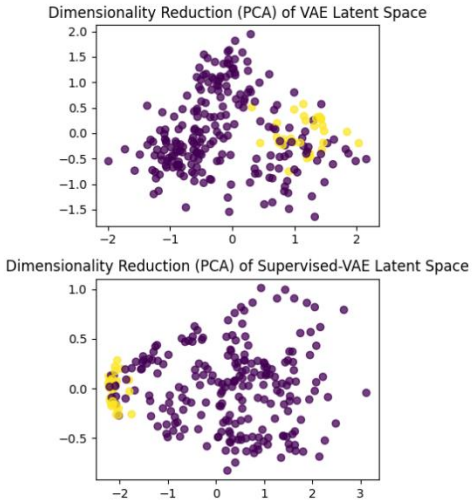


Figure 2b. Comparing Latent Representations



References

[1] Jack J Amend and Scott Spurlock. 2021. Improving machine learning fairness with sampling and adversarial learning. J. Comput. Sci. Coll. 36, 5 (January 2021), 14–23.

[2] Kingma, D.P. 2013. Auto-encoding variational Bayes. arXiv preprint. arXiv:1312.6114.

Illuminating Vulnerabilities - The Dark Side of Smart Bulbs

Jiayong Zheng, jzhen3561@ung.edu

Bryson Payne, bpayne@ung.edu

Department of Computer Science, Mike Cottrell College of Business

University of North Georgia, Dahlonega, GA

ABSTRACT

Smart devices, like smart bulbs, are devices that have the capability for users to remotely control their behavior and functionality. The ability for users to remotely control these devices also brings potential vulnerabilities that could sabotage an entire network. This research focuses on penetration testing of IoT devices to see how vulnerable these devices are to malicious contenders. IoT devices are daily-use appliances and tools that can connect to the internet. This includes the coffeemaker in some of our kitchens, garage door openers, even smart light bulbs that some of us use to remotely illuminate the room through our phone. Unlike desktop and laptop computers, these IoT devices rarely have any physical or logical intrusion detection or intrusion prevention mechanisms in them. Due to this, IoT devices are more prone to attacks compared to other wireless-compatible devices.

In this research project, the main goals to accomplish include intruding into a smart bulb, establishing remote connections with the bulb, and manipulating the smart bulb's behavior through scripting. The process starts with using Ubertooth One to capture the Bluetooth signal from the smart bulb. The ultimate command and control phase of the project is accomplished using scripts within a Kali Linux VM. Besides exploiting the vulnerabilities in the smart bulb, educating users on how to prevent such attacks from happening to their IoT devices is another focal point in this study. Defending IoT devices is feasible, and this paper concludes with recommendations for defense-in-depth strategies users can implement to protect their devices and home network.

PORTABLE DOOMSDAY DEVICES? FLIPPER ZERO VS. HACK RF IN MODERN HACKING

Ethan Lanio, Bryson Payne

Department of Computer Science and Information Systems
University of North Georgia
82 College Circle, Dahlonega, GA 30597
ellani6371@ung.edu, bryson.payne@ung.edu

Abstract

In the last decade, technical developments have made normal, commonplace devices into extremely potent devices, which is especially true within the hacking community. A new generation of portable hacking tools, like the HackRF One and Flipper Zero, have been created as our lives become more interconnected through smart gadgets and wireless networks. These tools let anyone to take advantage of weaknesses in a wide range of products, including car key fobs, garage doors, and much more. These small, easy-to-use gadgets are ushering in a new era where everyone and anyone can become a wireless hacker with the correct tools, as advanced hacking techniques become more accessible than before with a press of a few buttons.

In the field of portable wireless and hardware hacking, Flipper Zero and HackRF One are the most popular instruments with distinct features for penetration testers, security researchers, and enthusiasts alike. Designed for experimenting and hacking with RFID, NFC, sub-GHz, infrared, and GPIO pins, the Flipper Zero is a multipurpose tool that is very adaptable. Both novice and expert users will find it easy to use due to its small size and simple UI. As opposed to this, the HackRF One is a Software Defined Radio (SDR) that can both receive and transmit radio signals to which it can perform more sophisticated wireless assaults, such as spoofing and intercepting GSM, Bluetooth, Wi-Fi, and GPS communication protocols. This research contrasts these gadgets, emphasizing their technological variations and how they manage in today's hacking environment through testing both in performing multiple different attacks such as Side Channel attacks, BadUSB attacks, and Wi-Fi Deauth Attacks to list a few.

The emergence of hardware-based hacking tools has become a major worry for security experts and malicious actors alike in the quickly changing field of cybersecurity. The Flipper Zero and the HackRF are two gadgets that are receiving a lot of interest as of late. They are both strong and diverse platforms that let users learn about and engage in a wide range of wireless communication protocols. Despite being created for study and ethical hacking, these tools have grown in popularity among security experts, enthusiasts, and maybe even criminals. The security of these systems is critical as more businesses move toward networked systems and depend on wireless communication for daily operations. The Flipper Zero and HackRF are two separate yet complimentary technologies used in both offensive and defensive cybersecurity techniques.

The Flipper Zero is a portable, user-friendly gadget that lets users engage with many radio frequency (RF) protocols. It was created as a multi-tool for pen-testers and hackers alike. With its ability to read, write, and emulate protocols such as NFC, RFID, infrared, and sub-GHz communications, it's a great tool for beginners looking to get started with hardware hacking. In contrast, the HackRF is a more sophisticated Software-Defined Radio (SDR) that allows users to send and receive signals across a broad frequency range. Although the HackRF is more complicated to operate and requires more software configuration, cybersecurity researchers prefer it because of its greater customizability and wider range.

The Flipper Zero is frequently commended for its small size, simplicity of usage, and plenty of pre-configured features, which make it ideal for novices. Its lack of sophisticated customization possibilities and very short signal range, however, are its limitations. On the other hand, sophisticated users who need more control over signal processing and a wider frequency range might benefit from a more flexible platform offered by the HackRF. Although it offers an unmatched amount of control, its open-source nature and compatibility with potent SDR software suites like GNU Radio and SDR also come with a more challenging learning curve and the increased potential to harm as using the devices options with no consideration of its potential could lead to issues with law enforcement. In addition to this, the Hack Rf being used in this project is the Hack RF with portapack which gives it its on-the-go or portable nature versus the Hack RF One which needs to be connected via a computer.

This research projects examines the strengths and weaknesses of the Flipper Zero and the HackRF in relation to each other and contemporary hacking in general, as well as the ways in which each tool might be applied in different offensive contexts. We want to determine their efficacy in real-world applications by carrying out a number of useful tests or malicious attacks that can be done by both devices. More specifically, we will assess their usability and versatility in various hacking scenarios in addition to looking at how well they perform in popular assaults including side-channel attacks, signal interception, wireless protocol manipulation, etc. The results of this study will add to the growing body of knowledge on portable hardware hacking in the context of modern security, providing an attackers, either malicious or ethical, perspective on how these devices might be studied and used in practical settings.

Manipulating the CAN | A Multi-Vehicle Car-Hacking Comparison

Poster Session

Zachary Simmons

Dr. Bryson Payne
Department of Computer Science and Information Systems
University of North Georgia
Dahlonega, GA 30533
zmsimm8243@ung.edu

This research involves capturing and replaying packets that are going over different vehicles' Controller Area Network(CAN) systems and analyzing the differences between vehicles, the primary goal is to turn on certain features in the car like the blinkers from an external device and explore the real-world implications of replay attacks. A CAN injection, replaying the packets that are known to change some state within the car, is the attack used on the Toyota Rav4 to start the engine without the key from the headlights' connection to the CAN.

As seen by the Rav4 vulnerability, more attack vectors have emerged as vehicles increasingly become computers on wheels. While automotive manufacturers are constantly working on securing the vulnerabilities associated with vehicles' computer systems, keeping up with new attacks is a constant challenge for manufacturers. Since the advent of the wireless key in 1982, replay attacks have been an issue in the automotive industry, while rolling codes have mostly fixed this issue another attack, the relay attack, has emerged with the creation of the completely keyless vehicle keys. This underscores that as vehicle technology becomes more innovative so do the attacks that plague automotive manufacturers and customers.

Residential Investigation: Uncovering the Threats in Smart Homes Using Network Forensics

Edwin Denmark, Bryson Payne

University of North Georgia

ABSTRACT

Over the last decade, the use of Internet of Things (IoT) devices in residential areas has increased. Although IoT devices provide the general consumer with great convenience, IoT devices usually have low levels of security, leaving homes vulnerable to cyberattacks. In a typical environment, network forensic techniques are effective in detecting threats. However, these techniques may struggle to identify threats in a common IoT household. This research aims to evaluate how well current network forensic techniques can detect threats in residential IoT environments. By identifying the strengths and weaknesses of these tools, this project will help to enhance the security and reliability of IoT devices.

In order to test some forensic techniques, this research simulated malicious network activities by using a free packet manipulation software called Scapy. In order to create a controlled test environment that can replicate the everyday IoT devices people can find inside their homes, this study makes use of items such as smart bulbs, security cameras, and a smart TV. Wireshark and Zeek are some free open-source tools that are used within this study. These tools are used to capture network packets and investigate network traffic. The results will show the strengths of current forensic software in a typical household. By using these tools to effectively detect threats, the average consumer can find ways to minimize any attacks that utilize their devices.

Automated Conversion of Fill-in-the-blank and Short-answer Questions to Multiple-choice Questions

Danielle Mathieu (dmathieu@ggc.edu), Dylan Long (dlong12@ggc.edu)
Advisor: Dr. Wei Jin (wjin@ggc.edu)
Department of Information Technology, Georgia Gwinnett College, Lawrenceville, GA

1 Introduction

Multiple-choice questions (MCQ) can be automatically graded and due to this convenience, they are the most frequently used format for assessments. One drawback is that there are normally a small number of choices, most often four, for each question, which could make it easier for students to guess the correct answer. Ideally, the distractor choices should capture student misconceptions. Very often the choices are made by teachers without going through a rigorous process as in [1]. Teachers might fail to capture many student misconceptions.

The fill-in-the-blank or short-answer (FIB/SA) types of questions do not reveal any hints for students and are considered better ways to assess student learning than the multiple-choice type of questions. However, although correct answers can be specified for such questions in many course management systems, they often require manual grading due to variations in how students enter their answers. This makes it impractical to rely on these types of questions for regular assessment.

This project addresses these challenges by developing a software tool that can be used to convert an existing FIB/SA question into a corresponding MCQ using the answers students have given in previous semesters as the choices. The answers should have been graded carefully by instructors in order for this approach to work well. This will not only increase the number of choices for a MCQ making it more difficult for students to guess the correct answer, but also improve the assessment accuracy, since the choices are actual answers from students capturing their misconceptions.

2 Related Work

Creating effective MCQs is a vital part of assessing student learning. According to Caceffo et al. [1], developing a Concept Inventory (CI) involves carefully designing each question to address misconceptions that students commonly have. This process begins by analyzing student responses to open-ended questions to identify common mistakes and misunderstandings. These misconceptions are then used to create distractors — incorrect answer choices that reflect these misunderstandings. By including distractors based on actual student responses, the quality of MCQs is enhanced, as they mirror the thought processes of previous students. This approach helps create questions that challenge students' critical thinking and reduce the likelihood of guessing the correct answer.

Abreu et al. explore the effectiveness of implementing MCQs in programming education [2]. Their study found that MCQs can increase student engagement with the material while also enabling faculty to accurately assess students' understanding of programming concepts. Data was collected through performance tests and surveys, where students were asked to compare the use of coding questions versus MCQs. The study highlighted that to increase student engagement with difficult material, MCQs were necessary due to their immediate feedback and the ability to cover a broader range of topics.

3 Our Approach

A CSV file containing carefully graded anonymized student answers for a collection of FIB/SA questions was provided for this project. We developed a software tool that does the following: (1) Extracts questions from the given CSV file. (2) For each question, identify all the student answers and the points received by each answer. The collection of answers for a question will be choices for the generated corresponding MCQ. As a result, this tool allows for partial credit if professors have provided partial credit when grading previous student answers.
This ensures that the generated MCQs are reflective of real student responses and thus capture real student misconceptions, enhancing the relevance and effectiveness of the assessments.

Our program uses hash maps to store data as they allow for efficient organization and retrieval of data, making it easy to manage and access large sets of question-answer pairs quickly. For an answer that did not score 100% on a question, the answer and its score will be entered into a hash map called wrongAnswers. The keys for this map represent every unique FIB/SA question in the CSV file, while the value for each key is a list that holds the wrong answers and their scores. Correct answers are added to another hash map called correctAnswersMap. The keys for this map are also every unique FIB/SA question in the CSV file, with the value for each key being a list of correct student answers. Note that there might be multiple ways to get 100% for a question. The information captured by the two hash maps will be written to a CSV file in a proper format for multiple-choice questions that can be then uploaded directly into a D2L question bank or as a D2L quiz, where D2L is a course management system currently in use at our college. Note that the answer choices are shuffled to randomize the order. To ensure there are at least three wrong answers for every question, wrong answers such as "None of the above" and "All of the above" could be added. Figure 1 shows what an automatically generated MCQ looks like.

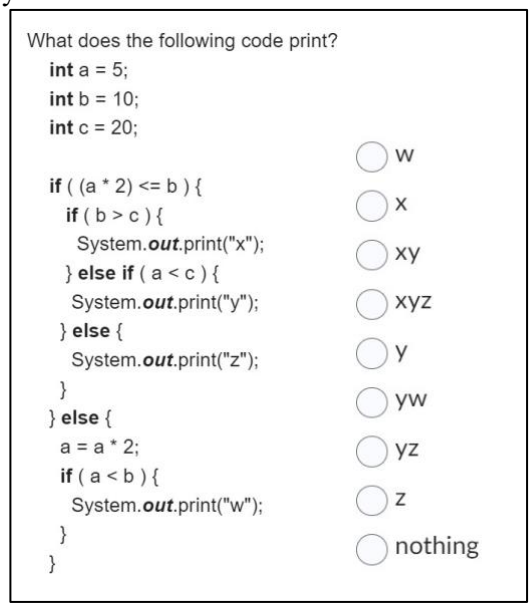


Figure 1: An Automatically Generated MCQ

4 Result and Conclusion

This tool was used successfully at the start of Fall 2024 for the efficient setup and grading of a Java preassessment for multiple sections of the Programming Fundamentals course. First, a professor exports student answers from a quiz into a CSV file. Multiple CSV files for the same quiz from different semesters or sections can be combined into one CSV file as the input to the program. The program will extract all FIB/SA question-answers and produce an output CSV file that contains the generated MCQs, which can be uploaded directly to D2L. Without this tool, it would not have been possible.

This project addresses the significant problem of manually grading FIB/SA questions, which is time-consuming and inefficient for professors. By developing a tool that automates the conversion of FIB/SA answers to MCQs, we provide a practical solution to streamline the assessment process. The tool ensures that answer choices are real answers previously provided by students, challenging new students to read and understand the questions more carefully. This not only saves valuable time for educators but also ensures consistency and fairness in grading, while reducing the likelihood of cheating. This automated solution significantly enhances the efficiency and effectiveness of creating and grading quizzes and serves as a valuable aid for professors.

REFERENCES

[1] Caceffo, R., Wolfman, S., Booth, K. S., & Azevedo, R. (2016, February). Developing a computer science concept inventory for introductory programming. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 364-369). DOI: https://doi.org/10.1145/2839509.2844559

[2] Abreu, P. H., Silva, D. C., & Gomes, A. (2018). Multiple-choice questions in programming courses: Can we use them and are students motivated by them?. *ACM Transactions on Computing Education (TOCE)*, *19*(1), 1-16. DOI: https://doi.org/10.1145/3243137

Generation of Equivalent Questions through Templates for Programming Concepts

Sean Nolan (snolan3@ggc.edu)
Advisor: Dr. Wei Jin (wjin@ggc.edu)
Department of Information Technology, Georgia Gwinnett College, Lawrenceville, GA

1 Introduction

The dissemination of assessment questions and answers over the internet (e.g. Chegg and Brainly) allows for students to receive credit for correct responses without properly testing their ability. One way to mitigate this issue is by creating different questions each semester to invalidate leaked assessments. However, manually creating new questions every semester becomes tedious and time-consuming, making it impractical for many professors. We present a tool that uses templates to automatically generate different versions of an assessment. This tool relieves the tedium of manually editing questions, making it effortless to create unique quizzes whenever necessary.

2 Background

Previous studies have been conducted to determine the impact of automatically generated unique quizzes on learning and teaching. Kumar developed online tutors for C++/Java programming [1]. The system uses templates to generate varied problems for the same concept. The main purpose of the tool is for tutoring. In addition to generating unique problems, the tutors grade the students' responses step by step and guide students to the final answer. The tutors can be adopted as an assessment tool. Currently, however, all the templates are part of the system and instructors cannot create their own questions.

Jin, Guo, et. al. experimented with template-based and automatically generated Chemistry quizzes [2]. Their project addressed a challenge associated with the mastery-based learning approach, which allows students to learn/relearn a concept until a certain level of mastery is achieved, measured by passing an exam for that concept. Since a student may take the exam multiple times, a different version of the exam will need to be generated for each attempt. Without a tool, instructors have to manually modify an existing quiz by changing numbers and units into an equivalent but different quiz, which is labor intensive and error prone. Data collected from a semester showed that, with the tool, the estimated total time saving per student is 354.8 minutes, based on roughly 10-minute time needed to manually generate each quiz variant. The saving could be enormous for large classes.

3 Our Template-based Approach

This project is based on the template-based approach as described above [2]. However, additional challenges needed to be addressed. Programming concept assessments often include code segments in question descriptions. The correct answer for such a question may not be captured by a formula. Our tool allows a code segment to be represented by a template. In addition, the tool can also generate variants for multiple-choice questions.

3.1 Question Template

To use this tool, a teacher will need to convert each quiz question into a template. A template consists of the specification of a set of template variables, the question itself, which has the template variables embedded in the question description, and answer description (see Figure 1). The tool can then generate random values for the template variables and replace variables' with these values in the question description and answers, resulting in a different version of the question.

A template variable can be of any data type. The specification of such a variable includes data type and how a value for this variable is generated. For a variable whose value is randomly generated, the range for the values must be specified. If a variable is of the string or the character type, the range is specified as a set of values that the tool can randomly pick from.

In addition, the *solution type* can be specified (e.g., the red box in Figure 1) and how solution is generated (e.g., the blue box in Figure 1). By default, the solution is determined by evaluating a mathematical expression. The second approach to generate the solution is by compiling and executing a given code segment. For multiple-choice questions, the solution generation method is also "compile & execute" as the choices will be printed by an embedded code. This allows for more control over the output of the program.

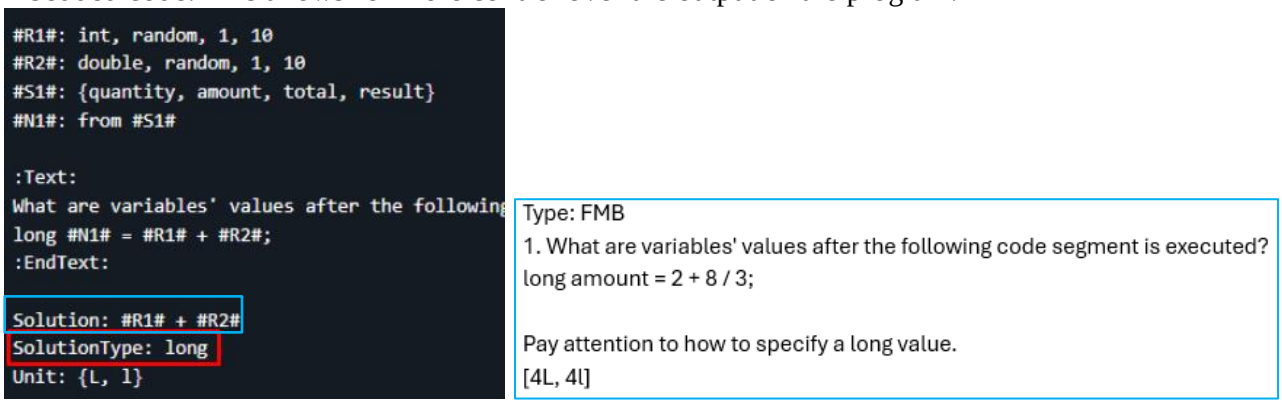


Figure 1: (a) An Example Template and (b) a Generated Question from the Template

3.2 Quiz and the Tool

A quiz is a text file that contains multiple questions, each of which is represented as a template described above. The tool parses a given quiz template file and processes each question template. For each question, it generates a random value for each template variable, embeds these generated values in the question text, and determines answers based on these values. Each execution of the tool will generate a unique quiz and outputs two files: One is a Word document to allow for reading of the generated assessment in a simple format, and the other is a CSV file for uploading to various online course management systems as a quiz or into the question bank.

4 Result and Conclusion

Without the tool, currently many quizzes/exams are either reused across semesters or have to be manually modified. To use this tool, a teacher will have to spend some initial time to set up a quiz template. After that initial time investment, each variant of a quiz can be instantly generated. Assuming that a course with 10 quizzes/exams a semester and 20 minutes time to modify each quiz into an equivalent variant, the time saved per semester per section is 10 * 20 = 200 minutes. For a course with multiple sections (e.g. 10 for the Programming Fundamentals course), the accumulated saving time could be substantial.

The development of a template-based, automated question generation tool offers a practical solution to the challenge of assessment integrity in educational settings. This tool saves valuable time for educators by automating the creation of exam variations and prevents cheating by rendering leaked assessments ineffective. With flexibility in terms of question types, variable replacements, and solution generation methods, it is suitable for use within a range of subjects and assessment styles. Therefore, this tool is a key milestone in the automation of assessment generation, given that institutions are in continuous pursuit of effective and dependable means of rating student learning.

REFERENCES

[1] Kumar, A. N. (2005, June). Online tutors for C++/Java programming. In *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education* (pp. 387-387). DOI: https://doi.org/10.1145/1067445.1067589

[2] Jin, W., Guo, Y., Marshall, D., & Brown, G. (2020, October). An Interdisciplinary Collaboration Enhances Education in Both Disciplines and Generates Future Opportunities. In *Proceedings of the 21st Annual Conference on Information Technology Education* (pp. 149-155). DOI: https://doi.org/10.1145/3368308.3415385

Project LegoLogic: Using Lego Spike Prime to Teach Essential Coding Concepts

Juan Guevara, Miguel Leon, Jenna Vincent, Dr. Cengiz Gunay, Dr. Cindy Robertson
Department of Information Technology, School of Science and Technology, Georgia Gwinnett College

Introduction

Technology has become an integral part of everyday life, and understanding its principles is essential in today's world. However, introducing complex topics like programming to young learners can be challenging. By using accessible and interactive tools, such as the LEGO Spike Prime system, we can make learning coding both engaging and fun for students.

This project introduces essential coding concepts through a hands-on workshop in which participants build and program models. Through this playful approach, we hypothesize that students will learn fundamental programming skills like loops, conditionals, and variables, gaining a solid foundation in coding and robotics. We will study whether we reach our target by surveying participants' knowledge in these workshops.

Study Target

This study is geared toward students in grades 6 and above who have minimal programming experience. This project aims to engage young students and spark their interest in technology at an early age. By teaching essential programming concepts through the interactive and playful use of LEGO Spike, the study aims to make learning coding both simple and enjoyable. The project also seeks to encourage collaboration among the participants with a fun, hands-on activity where students can create and play a game while gaining foundational coding knowledge.

Description of TAP program

The Technology Ambassadors Program (TAP) was created to spark an interest in information technology within the community using fun and interactive technology workshops [1]. TAP aims to engage a broad audience by developing projects that are simple enough to be understood by a wide range of participants, including both those with prior knowledge of information technology and complete beginners. This approach ensures that the program is accessible and appealing to people with varying levels of experience.

Methods

The project utilizes the LEGO Spike set [3] along with the LEGO Spike Word Block coding platform [2]. The LEGO Spike set includes various components such as motors and color sensors, which are essential for building interactive and dynamic models. The motors allow for movement and mechanical functionality, while the color sensors enable the detection of different colors, adding an element of responsiveness to the project. The LEGO Spike Word Block coding platform provides an intuitive drag-and-drop block coding interface, making it easy for beginners to learn and apply coding concepts without needing prior programming experience.

Through this project, the aim is to teach essential coding concepts, including loops, variables, and conditionals by constructing a catapult-like contraption that can be aimed to throw projectiles [4,5]. Participants will learn to create simple loops, such as "repeat" or "while" loops, to control motors and automate repetitive tasks. Additionally, participants will grasp the concept of conditionals, like "if-else" statements, using them to make decisions within their programs. This will involve programming the machine to make choices based on sensor input or predefined conditions, such as turning the catapult to aim it based on light sensor input. And finally, participants will learn how to create and use variables to store and modify data, including counters or sensor readings, throughout their program.

These fundamental concepts are the building blocks of programming, and by integrating them into a hands-on, interactive experience, the project makes learning engaging and accessible to students. We will conduct pre- and post-surveys to test the participants' knowledge gained on these topics in the workshop to test whether our method has been effective.

Figure 1. Catapult object being constructed with the LEGO Spike set.

Results

By the end of this LEGO Spike program lesson, participants will gain a foundational understanding of key programming concepts, including loops, conditionals, and variables, through the block coding provided by the LEGO Spike Prime system. The results of this study will be presented in detail in our accompanying poster, highlighting the learning outcomes and participant engagement throughout the lesson.

References

1. Dekhane, S., Xu, X., Napier, N., Barakat, R., Gunay, C., & Nagel, K. (2018). Technology focused service-learning course to increase confidence and persistence in computing. Journal of Computing Sciences in Colleges, 34(2), 147-153. https://dl.acm.org/doi/10.5555/3282588.3282609
2. Körei, A., & Szilágyi, S. (2022). From Scratch to Python : Lego Robots AS motivational tools for coding. *Multidiszciplináris Tudományok*, *12*(3), 247–255. https://doi.org/10.35925/j.multi.2022.3.22
3. *Lego Education Spike*, spike.legoeducation.com/prime/models/. Accessed 23 Sept. 2024.
4. *Spike Prime "Automatic Lego Catapult"*. Joy coding : 네이버 블로그. (n.d.). https://blog.naver.com/subeen40/223234159558
5. YouTube. (n.d.). *[Tutorial] The Ultimate LEGO Cannon Tutorial: Build, Aim, Fire!*. YouTube. https://www.youtube.com/watch?v=2pB8OkP1ReM

Acknowledgments

This project has been partially funded by the National Science Foundation grant #2315804. We would like to express our sincere gratitude to all those who contributed to the success of this project. We also extend our appreciation to the GGC Technology Ambassadors Program (TAP) faculty and students and the School of Science and Technology for providing the resources and platform for this project.

Building A Pokémon Chatbot Using Retrieval Augmented Generation

Jonathan Tran, Dr. Cengiz Gunay
Department of Information Technology, School of Science and Technology, Georgia Gwinnett College

Introduction

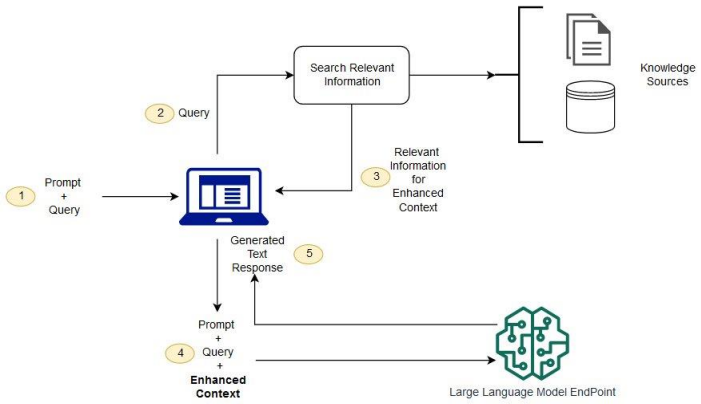
Artificial Intelligence (AI) is a field of study that allows computers to perform complex and advanced functions based on the user's preference. The first few recorded instances of AI came out in the 1950s, when Alan Turing invented the Turing test for machine intelligence as it was popularized in the movie (The Imitation Game [1]), theorizing the general idea of whether machines can think like humans. Recent advances in AI came close to this vision with Large Language Models (LLM) [2]. LLMs are a subset of AI that primarily involves recognizing and generating text for that specific task. They are composed of essentially a set of neural networks that generalizes data and matches the meaning of that specific data to its own pre-trained data. In other words, LLMs are models that can understand meanings and relationships of written texts. In this project, we are applying LLMs to create a chatbot that responds to queries about Pokémon.

At the time of writing, Pokémon is roughly 26 years old, having existed within 2 generations, which are Millennials and Gen Zs. There is a high probability that if you were one of these generations, you know of Pokémon and have seen some form of its media. Being wildly popular, Pokémon media types produced include video games, trading card games, or animated cartoons, resulting in massive amounts of media on it ranging from multiple shows and video game iterations to its trading card counterpart. Despite the large fan base, information pertaining to anything Pokémon related is scattered across libraries for Pokémon that exist on the internet. Furthermore, most of these resources are incredibly hard to understand as they contain too many small details, which makes it difficult for the average person to navigate them. We propose that a chatbot that does the searching would make this process much more efficient and save people a massive amount of time.

We will train the chatbot with Pokémon data, which will include all generations, 1 through 9. Each generation is marked by the iteration of its corresponding game. For example, generation 1 refers to the first 3 Pokémon games, which are Pokémon Red, Blue, and Yellow. Generation 2 then refers to the set of games following them. This is the typical pattern when it comes to naming Pokémon generations. In the total of nine generations, there are 1026 separate Pokémon data entries. Each entry will have its own stats, location of the entry, type weaknesses and strengths, etc. We will feed this data in natural language to the AI model by using the following chatbot implementation.

Methods

To create a working chatbot that is well versed in Pokémon data, we used a web API, Poke Data [3], that contains all necessary data. The code used various Python libraries to query and display the chatbot. Some of these libraries include Streamlit [4], Ollama [5], Langchain [6], and others. Streamlit [4]is used because it allows the quick creation of a web-based app, Langchain [6] is used to connect large language models into applications, and Ollama [5] is used as the main LLM in which to train our data. Specifically, the chatbot will be running Facebook's llama 3.1 model [7].



Integrating the Pokémon data into a general-purpose LLM can be achieved by using the Retrieval Augmented Generation (RAG) [8]. This works by feeding into the LLM pre-trained data as context, which will then be vectorized, or split apart into smaller chunks, and stored. After the data is vectorized, the user will send a query, and the model will find the closest match from the query to the data using text classification. Figure shows the workflow of RAG (Refer to [1] for image source).

Expected Results

This project will be showcased at a local symposium where users can view and potentially play with the chatbot. They could help test any overlooked errors that got past initial testing (e.g. query is too specific so the chatbot is unable to give an accurate response). The chatbot's creation is completed but will require minor changes to the code for the updated data. We will present our results on our poster.

References

[1] A. M. Turing, "I.-Computing machinery and intelligence," OUP Academic, https://academic.oup.com/mind/article/LIX/236/433/986238?login=false#164226500 (accessed Sep. 29, 2024).

[2] "The history, timeline, and future of llms," Toloka, https://toloka.ai/blog/history-of-llms/ (accessed Oct. 5, 2024).

[3] "Documentation," PokéAPI, https://pokeapi.co/docs/v2 (accessed Sep. 29, 2024).

[4] "API reference - streamlit docs," API Reference - Streamlit Docs, https://docs.streamlit.io/develop/api-reference (accessed Sep. 29, 2024).

[5] Ollama, "Ollama/docs/api.md at main · ollama/ollama," GitHub, https://github.com/ollama/ollama/blob/main/docs/api.md (accessed Sep. 29, 2024).

[6] "Introduction," LangChain, https://python.langchain.com/v0.2/docs/introduction/ (accessed Sep. 29, 2024).

[7] Meta-Llama, "Meta-llama/llama-models: Utilities intended for use with Llama models.," GitHub, https://github.com/meta-llama/llama-models (accessed Sep. 29, 2024).

[8] "Rag," RAG, https://huggingface.co/docs/transformers/en/model_doc/rag (accessed Oct. 5, 2024).

[9] What is Rag? - retrieval-augmented generation AI explained - AWS, https://aws.amazon.com/what-is/retrieval-augmented-generation/ (accessed Sep. 30, 2024).

A Structural Analysis of Largest Independent Sets in a Family of Circulant Graphs

Extended Abstract

Moazzam Hayat

September 2024

1. Introduction and Background

Our goal is to prove an upper bound on $\alpha(G)$, the independence number, i.e., the size of a largest independent set, for each graph G in a particular infinite family of circulant graphs. For general graphs, the problem of determining the independence number is NP-complete [GR01]. The problem has not been solved for circulant graphs either [Sto14]. As discussed below, in an optimal assignment of frequencies to a set of wireless devices, our efforts will place an upper bound on the minimum separation of frequencies assigned to devices within close proximity of each other.

The *Frequency Assignment Problem (FAP)* involves assigning frequencies to wireless devices in a region so that, to avoid interference, devices close to each other have distinct frequencies. Moreover, it is desirable that the minimum separation of frequencies assigned to neighbouring devices, denoted s_1 below, be as large as possible. Devices that are beyond a certain *re-use distance*, σ, from each other could be assigned the same frequency. An *optimal* assignment uses as few frequencies as possible.

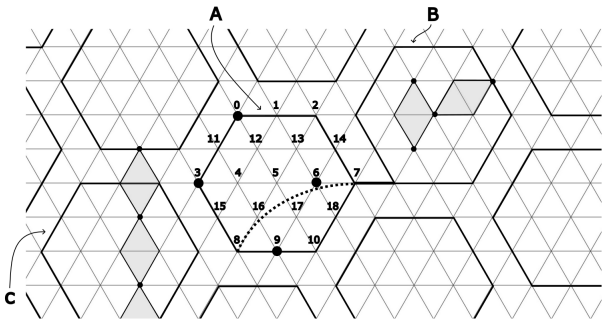


Figure 1: Tessellation for $k = 2$ with copies of H_2; a LIS in H_2; diamonds and chains of diamonds

Shashanka et al model FAP as a graph colouring problem (using natural numbers $N_m = \{0, 1, \ldots, m\}$[1] for frequencies/colours) for the infinite graph, $\mathcal{A}_2$, corresponding to the unit equilateral triangular grid, where the vertices (edges) of the triangles are the vertices (edges) of the graph [SPS03]. They consider the case where vertices at distance less than σ have distinct colours. They show that, given $\sigma = k+1$, (i) the infinite grid graph can be tessellated by hexagons of radius k, (ii) a colouring of the graph, $H_k = (V_k, E_k)$, induced by a hexagon along with wrap-around edges on the borders (corresponding to edges from the border of one hexagon to the border of a neighbouring hexagon in the tessellation) needs at least as many colours as the number of vertices, $|V_k| = 3k^2 + 3k + 1$, in H_k, i.e., $m \geq |V_k| - 1$, (iii) an optimal colouring of H_k is an optimal colouring for the infinite grid graph, and (iv) there exists an optimal colouring for H_k such that

[1]Note that the set N_m is cyclic, i.e., difference between 0 and m is 1.

$s_1 = k^2$. For $k = 2$, Figure 1 shows the tessellation, and the hexagon labelled A is H_2 with one of the wrap-around edges. Hexagon A also shows a largest independent set (LIS) in this copy of H_2.

For a given k, consider an optimal colouring of H_k that achieves $s_1 = t$. Clearly, then, the set of vertices assigned colours in the set $\{0, \ldots, t-1\}$ is an independent set in H_k. In the assignment shown by Shashanka at al, $s_1 = k^2$. Consequently, $\alpha(H_k) \geq k^2$. If $\alpha(H_k) \leq k^2$, then the assignment shown by Shashanka et al maximises s_1. Hence our interest in finding an upper bound on $\alpha(H_k)$.

2. Our Approach

We label the vertices of H_k such that for any vertex labelled x, the neighbors of x are labelled $(x + i) \mod |V_k|$ where $i \in N_k$ and $N_k = \{1,\ 3k + 2,\ 3k + 1,\ -1,\ 3k^2 + 1,\ 3k^2\}$. Thus, H_k is a circulant graph. Henceforth, we will refer to vertices of H_k by their labels.

For a vertices $v, u \in V_k$, u is defined to be a *diamond point of* v, iff $d(v, u) = 2$, and in a shortest path from v to u, $v \xrightarrow{i_1} v' \xrightarrow{i_2} u$, $v' = (v + i_1) \mod |V_k|$, $u = (v' + i_2) \mod |V_k|$, and $i_1 \neq i_2 \in N_k$. We will denote u by $dp(v, i_1, i_2)$. The structure formed between v and $dp(v, i_1, i_2)$ is called a *diamond*, denoted by $D_k(v, i_1, i_2)$. Each diamond has an *orientation*, and two diamonds, $D_k(u, i_1, i_2)$ and $D_k(v, j_1, j_2)$ are said to be in the same orientation iff in the shortest paths $i_1 = j_1$ and $i_2 = j_2$, or $i_1 = j_2$ and $i_2 = j_1$. Hence, a *chain of diamonds* is defined as the sequence of distinct vertices $v_0, \ldots, v_n$ iff there exist $i_1, i_2 \in N_k$ such that for all $x < n$, $dp(v_x, i_1, i_2) = v_{x+1}$. Note that all the diamonds in a chain of diamonds, by definition, have the same orientation. The hexagons labelled B and C in Figure 1 show examples of diamonds and a chain of diamonds, respectively.

It can be easily shown that starting at vertex 0, we can follow the chain of diamonds at subsequent vertices in the same orientation, and this chain visits each vertex in H_k, and after $|V_k|$ vertices is back at the vertex 0. We will call the chain traversed in the vertical orientation as a *complete chain of diamonds for* H_k.

With a careful study of the structure of a largest independent set (LIS) S in H_k, we have shown the following two results.

Lemma 1. *For any LIS S in H_k, for any vertex v in S, there exists another vertex v' in S that is a diamond point of v.* □

Lemma 2. *If all the vertices in a chain of diamonds belong to a LIS S in H_k, then the chain has at most k vertices.* □

We are currently in the process of proving

Conjecture 1. *Any LIS S in H_k can be procedurally transformed to a LIS S' such that S' is the union of mutually disjoint sets S_i, and for each i, the vertices of S_i form a chain of diamonds in the vertical orientation.*

Sketch of proof (work in progress) Our transformation procedure runs in rounds. In each round, we traverse the complete chain of diamonds for H_k in sequence, starting at vertex 0, and for each vertex, ensure that v has a diamond point in the vertical orientation. For any vertex v in a LIS S, we have tabulated every possible configuration of LIS vertices in the neighbourhood of v, and shown that, in each case, either v has a diamond point in the vertical orientation, or, when possible, we can move vertices of S so that the resulting set is a LIS and v has a diamond point in the vertical orientation. Our movement of vertices is restricted to vertices that appear after v in the complete chain of diamonds. We are in the process of showing that after a bounded number of rounds, all the vertices in the resulting LIS will have diamond points in the vertical orientation, thus proving our conjecture. □ Sketch

Then, using Lemma 2 and Conjecture 1, we will try to show that

Conjecture 2. *Suppose S, a LIS in H_k, is the union of mutually disjoint sets S_i, and for each i, the vertices of S_i form a chain of diamonds in the vertical orientation, then, $\sum_i |S_i| \leq k^2$.*

The proofs of Conjectures 1 and 2 together will prove that $\alpha(H_k) \leq k^2$.

References

[GR01] Chris Godsil and Gordon Royle. *Algebraic Graph Theory*. Springer, 2001.

[SPS03] M. V. S. Shashanka, A. Pati, and A. M. Shende. "A characterisation of optimal channel assignments for wireless networks modelled as cellular and square grids". In: *Proceedings International Parallel and Distributed Processing Symposium* (2003), p. 8. DOI: 10.1109/IPDPS.2003.1213406.

[Sto14] Derrick Stolee. "Finding Cliques and Independent Sets in Circulant Graphs". In: *Computational Combinatorics* (2014).

Is AI Coming for Our Jobs as TAs and Academic Advisors?

Emirhan Gencer, and Jack Patterson
Furman University

This project explores the development and challenges of creating AI chatbots for academic support roles, such as teaching assistants (TAs) and advisors. We discuss the process of data collection and scraping, vectorization techniques, and the utilization of Large Language Models (LLMs) like OpenAI's GPT. Key challenges addressed include data cleaning and formatting, cost reduction strategies, maintaining chat history relevance, and restricting the domain of questions without hindering the chatbot's functionality. We also delve into the complexities of incorporating visual explanations and mitigating biases in AI models. Through our work, we aim to shed light on the potential of AI in revolutionizing academic support while highlighting the intricacies and considerations involved in its development.

Participation and Success of Underrepresented Students in Computer Science

Nishyah Scott and Michael Diaz
Furman University

This work explores factors influencing participation and success in computer science, focusing on underrepresented students. It leverages IPEDS, Taulbee Survey, College Scorecard, and Furman University data to examine enrollment, retention, performance, and earnings. Key findings emphasize the importance of representation among faculty and PhD students for encouraging undergraduate participation from similar backgrounds. Computer science proves accessible to low-income students, offering a potential path to upward mobility. However, gaps in high school GPA and quality persist, impacting performance. The research also highlights a concerning trend of students, particularly Black and first-generation students, intending to major in computer science but not completing the program. This attrition often occurs early in the academic journey, potentially linked to lower GPAs. The presentation concludes with a call for qualitative research to contextualize these quantitative findings and to explore student and faculty perspectives on factors contributing to the lack of diversity and inclusion in computer science.